## Processes

There are various processes going on between the application and the database as well as the user and the application. In this section, the processes are presented and analyzed.

The main process is the process which takes care of the main function of the application. The optional features are the functionalities which the user may choose to utilize or not by clicking a button.

The processes are classified into three categories:

1. User ↔ Application: mainly concerned with GUI processes and how the user is notified.
2. Application **→** Database: mainly concerned with what the application requests from the database and how the application handles retrieved data internally.
3. Application **←** Database: mainly concerned with what the database is doing in response to the application’s requests.

### Authentication

#### Main process

Before accessing any of the provided applications, the user has to log in with their credentials.

The process is as follows:

|  |  |  |
| --- | --- | --- |
| **Steps** | **Category** | **Details** |
| **1** | User ↔ Application | User enter credentials |
| **2** | Application **→** Database | If the username is “admin”, query the database |
| **3** | Application **←** Database | Return number of rows with matching username and password |
| **4** | Application **→** Database | If the row number is 1, then the credentials are valid. If not, the credentials are invalid |
| **5** | User ↔ Application | User is notified of the result |

*Figure 1. Authentication process*

If the user logs in successfully, the main form of the application will be displayed. Otherwise, the user remains in the login form.

### Checking in to the event

#### Main process

To successfully check in, the user has to first scan their QR code. The QR will have the ticket number encoded in it.

The scanning QR process is as follows:

|  |  |  |
| --- | --- | --- |
| **Steps** | **Category** | **Details** |
| **1** | User ↔ Application | User press button to start QR scanner (a webcam, in this case) |
| **2** | Application **→** Database | The application turns on webcam |
| **3** | User ↔ Application | User positions their QR code against the scanner |
| **4** | Application **→** Database | The application tries to decode the QR |
| **5** | User ↔ Application | A message is displayed accordingly |

*Figure 2.1 Scanning QR process*

If a QR is found, the ticket validation process is as follows:

|  |  |  |
| --- | --- | --- |
| **Steps** | **Category** | **Details** |
| **1** | Application **→** Database | Extracts ticket number from QR. Queries the database for ticket’s information |
| **2.1** | Application **←** Database | Returns ticket information with the given number |
| **2.2** | Application **→** Database | Registers ticket to intermediate Ticket object |
| **2.3** | Application **→** Database | Queries database for the Visitor with said ticket |
| **2.4** | Application **←** Database | Return visitor’s information |
| **2.5** | Application **→** Database | Registers visitor’s information to intermediate Visitor object |
| **3.1.1**  **(for paid tickets)** | User ↔ Application | Application displays ticket and statuses:   * Ticket not found: NOK * Not paid: NOK * Paid: OK |
| **3.1.2** | User ↔ Application | User scans RFID card to link |
| **3.1.3** | Application **→** Database | Registers data to intermediate Visitor object. Updates the database. |
| **3.1.4** | Application **←** Database | Register ticket’s entry time Registers visitor’s RFID and status as checked in |
| **3.2.1**  **(for unpaid tickets)** | User ↔ Application | User can choose to change the ticket’s status |
| **3.2.2** | Application **→** Database | Registers data to intermediate Ticket object. Updates the database. |
| **3.2.3** | Application **←** Database | Register tickets as paid |
| **3.2.4** | User ↔ Application | Application displays retrieved data onto labels |

*Figure 2.2 Ticket validation process*

#### Optional features

The application responsible for checking in also offers the following features:

1. View visitor’s details.

This was made optional due to the fact that in most cases, checking in should be successful. There would not be a constant need of displaying the visitor’s personal details. Should they choose to view the information, the information is retrieved from the visitor’s object and displayed.

1. Check history

If the ticket has already been used, there is an option to check for any transactions made by the owner of said ticket. This feature is implemented for control purposes.

|  |  |  |
| --- | --- | --- |
| **Steps** | **Category** | **Details** |
| **1** | User ↔ Application | User presses button |
| **2** | Application **→** Database | Query for number of orders made by the visitor |
| **3** | Application **←** Database | Return number of records found |
| **4.1** | User ↔ Application | User sees QR code is not in use |
| **4.2** | User ↔ Application | User sees QR code is in use |

*Figure 2.3 Process of checking visitor’s history*

This feature is reserved for supervisors. When overriding, the user is required to go through authentication again, this time with more restrictions.

When logged in as admin, scanning of any ticket will result in a successful operation.

|  |  |  |
| --- | --- | --- |
| **Steps** | **Category** | **Details** |
| **1** | User ↔ Application | Admin user scans ticket |
| **2** | Application **→** Database | If ticket is unpaid, change status to ensure a successful check-in. Updates intermediate object. Updates database. |
| **3** | Application **←** Database | Registers new ticket’s status. |
| **4** | User ↔ Application | Application displays status as OK |

*Figure 2.4 Overriding check-in process*

### Checking out of the event

#### Main process

The checking out process is as follows:

|  |  |  |
| --- | --- | --- |
| **Steps** | **Category** | **Details** |
| **1** | User ↔ Application | User scans RFID |
| **2.1** | Application **→** Database | Queries the database for visitor’s information with said tag |
| **2.2** | Application **←** Database | Return visitor’s information |
| **2.3** | Application **→** Database | Registers data to intermediate Visitor object |
| **2.4** | Application **→** Database | Queries the database for unreturned items |
| **2.5** | Application **←** Database | Return records of unreturned items (if any) |
| **2.6** | Application **→** Database | The application registers the items into a List<LoanArticle> in the Visitor object |
| **3.1.0** | Application **→** Database | The application evaluates the list of items |
| **3.1.1** | Application **→** Database | The application tells database to check out the visitor |
| **3.1.2** | Application **←** Database | Move visitor’s record to deleted\_visitor table |
| **3.1.3** | User ↔ Application | User see status “OK” |
| **3.2.1** | User ↔ Application | User see status “NOK” together with item’s information |

*Figure 3.1 Checking out process*

#### Optional features

The application responsible for checking out also offers the following features:

1. Override

This feature is reserved for supervisors. When overriding, the user is required to go through authentication again, this time with more restrictions.

When logged in as admin, scanning of any RFID will result in a successful check out.

|  |  |  |
| --- | --- | --- |
| **Steps** | **Category** | **Details** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User → Application** | **Application → Database** | **Application ← Database** |
| **Step 1** | Admin user scans ticket |  |  |
| **Step 2** |  | Registers intermediate Visitor object as checked out.  Updates database. |  |
| **Step 3** |  |  | Move visitor’s record to deleted\_visitor table |
| **Step 4** | User sees “OK” status |  |  |

*Figure 3.2 Overriding check-out process*

### Loaning items

**Main process**

To check out, the RFID card of the visitor must first be scanned.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User ↔ Application** | **Application → Database** | **Application ← Database** |
| **Step 1** | User scans RFID card |  |  |
| **Step 2** |  | Queries database for visitor with said tag. |  |
| **Step 3** |  |  | Return visitor’s information |
| **Step 4** |  | Registers information to intermediate Visitor object |  |

*Figure 4.1 Process of scanning RFID to loan*

After which, a message box will be shown to notify the user that they can now scan the items. The process behind the scanning items to loan is:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User ↔ Application** | **Application → Database** | **Application ← Database** |
| **Step 1** | User scans item |  |  |
| **Step 2** |  | Queries database for item with said tag. |  |
| **Step 3** |  |  | Return item’s information |
| **Step 4** |  | Registers information to intermediate object. |  |
| **Step 5** | User picks return date |  |  |
| **Step 6** | User presses “Confirm” button once all items are scanned |  |  |
| **Step 6** |  | Evaluates the deposit values of all items scanned |  |
| **Step 6.1** (Not enough credit on card) | User is notified |  |  |
| **Step 6.2.1** (Sufficient credit) |  | Credit is being deducted from intermediate object.  Formulates SQL to   * Add order * Mark items as loaned * Deducts visitor’s credit   Queries the database with said SQL. |  |
| **Step 6.2.2** |  |  | Adds new order record  Registers the items as not available  Deducts visitor’s credit |
| **Step 6.2.3** | User sees “Loan successful!” status |  |  |

*Figure 4.2 Process of scanning items to loan*

**Optional features**

* Remove

User can also choose to remove the scanned items out of their loaning list.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User ↔ Application** | **Application → Database** | **Application ← Database** |
| **Step 1** | User selects items in the list, clicks “Remove” button |  |  |
| **Step 2** |  | Removes the item from the intermediate object. |  |

*Figure 4.3 Process of removing items out of loaning list*

### Returning items

**Main process**

To return the items, the user must first scan their RFID card – a process of which is described in Figure 4.1.

After which, a message box will be shown to notify the user that they can now scan the items. The process behind the scanning items to return is:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User ↔ Application** | **Application → Database** | **Application ← Database** |
| **Step 1** | User scans item |  |  |
| **Step 2.1** |  | Queries database for item with said tag. |  |
| **Step 2.2** |  |  | Return item’s information |
| **Step 2.3** |  | Registers information to intermediate object. |  |
| **Step 3** |  | Calculate credits to return/deduct based on price and deposit value. |  |
| **Step 4** |  | Formulates SQL to:   * Mark item as returned * Change visitor’s credit   Queries the database. |  |
| **Step 5** |  |  | Items are registered as available.  Visitor’s credit is changed. |
| **Step 6** | User sees status signifying successful return.  User is notified with the item’s information as well as credits being returned/deducted. |  |  |

*Figure 5 Process of scanning items to return*

### Selling items

To sell items, the application must firs load the available items from the database.

Since the application also supports the loading of pictures onto appropriate buttons, the application prepares for this by:

* The buttons which will hold the images are named with a prefix “productButton”
* Initialize a list of buttons to load the pictures on
* Cycle through all available buttons in the application and add the ones with the prefix “productButton” to the list

A similar process is done to the labels.

By now, the application will have a way of accessing the appropriate labels and buttons to load the information to.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User ↔ Application** | **Application → Database** | **Application ← Database** |
| **Step 1** |  | Queries database for all available items |  |
| **Step 2** |  |  | Return item’s information |
| **Step 3.1** |  | Store items in intermediate list |  |
| **Step 3.2** |  | Sort list by Category |  |
| **Step 3.3** |  | Extract image’s filename from the items and load it to the button  Change button’s background color based on Category. |  |
| **Step 3.4** |  | Extract item’s name and load it to the label |  |

*Figure 6.1 Process of loading item’s information to application*

The user can then click on a button to add it to their order.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User ↔ Application** | **Application → Database** | **Application ← Database** |
| **Step 1** | User presses item’s button |  |  |
| **Step 2** |  | Extract proper index based on the button’s name |  |
| **Step 3** |  | Update intermediate order object |  |

*Figure 6.2 Process of adding items to order*

There is also the option to add or remove the quantity – one which functions similar to the remove feature in Figure 4.3.

After the user is certain with their order, they can press the “Confirm” button to finalize the payment. On press, the application will await a scan of the RFID.

When the card is scanned, the following process is carried out and the order is marked as complete.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User ↔ Application** | **Application → Database** | **Application ← Database** |
| **Step 1** | User scans RFID card |  |  |
| **Step 2** |  | Query database for visitor with said tag |  |
| **Step 3** |  |  | Return visitor’s information |
| **Step 3.1** |  | Register information to intermediate visitor object |  |
| **Step 4** |  | Deduct credit to visitor object.  Formulate SQL to handle order. |  |
| **Step 5** |  | Query database |  |
| **Step 6** |  |  | Deduct item’s quantity  Deduct credit  Add a new order record |
| **Step 7** | User is notified of successful order |  |  |
| **Step 7.1** |  | Intermediate objects are reset for new order |  |

*Figure 6.3 Process of handling order*